Base de dados
Resumos

Rodrigo Santos
Universidade NOVA de Lisboa

1 Diagramas entidade e relação (DER)

Retângulos: representam conjuntos de entidades.
Losangos: representam conjuntos de relações.
Elipses: representam atributos. **Vamos fazer as coisas por forma a ter sempre atributos simples, uni-valor e não derivados**:

- Atributo **simples**.
- Atributo **composto**: Composto por vários atributos simples.
- Atributo **uni-valor e multi-valor**: exemplo de um atributo multi-valor: número**s** de telefone.
- Atributo **derivado**: podem ser calculado a partir de outros atributos, como por exemplo a idade, a partir da data de nascimento.

Linhas: ligam atributos aos conjuntos de entidades e conjuntos de entidades a conjuntos de associações.
Seta ($\rightarrow$): significa 1.
Linha (—): significa muitos.
Participação total (=): indicado por uma linha dupla, significa que toda a entidade do conjunto de entidades participa em pelo menos uma relação do conjunto de relações.
Participação parcial: algumas entidades podem não participar em qualquer relação do conjunto de relações.
Conjunto de entidades fraco: Um conjunto de entidades fraco é representado por um retângulo duplo. O atributo discriminante do conjunto de entidades fraco é sublinhado a tracejado. As relações são representadas por um retângulo duplo (a relação entre o conjunto fraco e o dominante).

Especialização/Generalização (ISA):

- **Herança de atributos**: um conjunto de entidades de menor nível herda todos os atributos e participa em todas as relações do conjunto de entidades de maior nível. Por outras palavras herdam todos os atributos dos que estão acima deles, e por isso não necessitamos de voltar a repetir os atributos.
- **Disjunta**: só pode pertencer a um elemento do nível inferior. Representa-se com um **disjoint** ao lado do triângulo do ISA.
- **Sobreposição**: pode pertencer a vários elementos do nível inferior.
- **Total**: tem de pertencer a pelo menos um elemento do nível inferior. Representa-se com um linha dupla (Como a Participação total.)
- **Parcial**: pode não pertencer a nenhum.

Chaves:

- **Super-chave de um conjunto de entidades**: é um conjunto de um ou mais atributos cujos valores determinam univocamente cada uma das entidades dentro do conjunto. Uma super-chave pode ter informação desnecessária.
- **Chave candidata**: Uma chave candidata de um conjunto de entidades é uma super-chave minimal.

- **Chave primaria**: Chave primária é uma chave candidata designada para identificar as entidades dum conjunto.
- A combinação das chaves primárias dos conjuntos de entidades participantes formam uma super-chave do conjunto de relações. **isto significa que um par de entidades pode aparecer no máximo uma vez num conjunto de relações**.
- Temos que considerar a cardinalidade de mapeamento (Restrições de mapeamento mais acima) dos conjuntos de entidades quando decidimos quais as chaves candidatas dos conjuntos de relações.

2 Modelo relacional

2.1 Derivação a partir de um DER

Uma base de dados que seja representável por um DER pode ser também representada por intermédio de várias relações (tabelas). Para cada conjunto de entidades (retângulo) e para cada conjunto de relações do modelo ER (losango) gera-se uma única relação (tabela) com o nome do conjunto de entidades ou conjunto de relações respetivo.

Derivação de conjuntos de entidades como tabelas

Um **conjunto de entidades forte** reduz-se a uma relação (tabela) com exatamente os mesmos atributos. Um **conjunto de entidades fraco** é representado por uma relação (tabela) que tem como atributos a chave primária dos conjuntos de entidades dominantes (ou identificadores) e os restantes atributos do conjunto de entidades fraco.

Derivação de conjuntos de relações como tabelas

Um **conjunto de relações** é representado por uma relação (tabela) com atributos para as chaves primárias dos conjuntos de entidades participantes e com atributos adicionais para os atributos próprios (ou descritivos) do conjunto de relações.

Derivação de tabelas para a especialização (ISA)

Formar uma relação (tabela) para a entidade de maior nível (mais geral). Criar uma relação (tabela) para cada conjunto de entidades de nível abaixo, incluindo a chave primária da entidade acima e os atributos locais.

Derivação da Agregação

Tratar o conjunto de relações agregado como se se tratasse de um conjunto de entidades, sendo a sua chave a chave do conjunto de relações (as chaves primarias dos envolvidos).

2.2 Determinação de chaves a partir do DER

- **Conjunto de entidades forte**: A chave primária do conjunto de entidades é a chave primária da relação (tabela).
- **Conjunto de entidades fraco**: A chave primária da relação (tabela) consiste na união da(s) chave(s) primária(s) do(s) conjunto(s) de entidades dominante(s) com o discriminante do conjunto de entidades fracas.
- **Conjunto de relações**: A união das chave primárias dos conjuntos de entidades relacionados é uma super-chave da relação (tabela).
  - Para conjuntos de relações binários um-para-muitos, a chave primária do lado "muitos" é a chave primária da relação (tabela).
  - Para conjuntos de relações um-para-um, a chave primária de qualquer um dos conjuntos de entidades é chave candidata da relação (tabela).
  - Para conjuntos de relações muitos-para-muitos, a união das chaves primárias é a chave primária da relação (tabela).

2.3 Tabelas redundantes

Conjuntos de relações **muitos-para-um** e **um-para-muitos**, totais no lado muitos podem ser representados adicionando atributos extra ao lado *muitos* contendo a chave primária do outro conjunto participante.
Se a participação é parcial (—) no lado muitos, a substituição da relação (tabela) por atributos extra pode levar à ocorrência de valores nulos.
Para conjuntos de relações **um-para-um**, qualquer dos lados com participação total **(=)** pode receber a chave primária do outro lado.
São redundantes as relações (tabelas) correspondentes aos conjuntos de relações entre o conjunto de entidades fracas e os seus conjuntos de entidades dominantes.

2.4 Chaves Estrangeiras

Um esquema de relação pode ter um (ou mais) atributo(s) que corresponda(m) à chave primária de outra relação. Esses atributos são designados por chaves estrangeira. As chaves estrangeiras impõem restrições sobre os dados (integridade referencial). Apenas os valores que ocorrem na relação referenciada podem ocorrer nos atributos da chave estrangeira da relação referenciadora.

3 Álgebra Relacional

3.1 Operação de seleção

$$\sigma_p(r) = \{t \mid t \in r \wedge p(t)\}$$

Onde p é o predicado de seleção, que é uma formula proposicional com termos ligados por: $\wedge, \vee, \neg$. Cada termo é do tipo $< \mathit{Atributo} > \ \mathit{op} \ < \mathit{Atributo} >$ onde *op* pode ser $=, \neq, >, \geq, <, e \leq$.
Propriedades de seleção:

- *Comutatividade* — $\sigma_1(\sigma_2(r)) = \sigma_2(\sigma_1(r)) = \sigma_{1 \wedge 2}(r)$.

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

Table 1: Relação r

A	B	C	D
α	α	1	7
β	β	23	10

Table 2: $\sigma_{A=B \ \wedge \ D>5}(r)$

3.2 Operação de projeção

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

Onde $A_1, A_2, \ldots, A_k$ são nomes de atributos e r é uam relação. O resultado é a relação com os k atributos (k colunas) obtidos eliminando os atributos (colunas) que não estão listadas. **O resultado não tem tuplos duplicados, pois as relações são conjuntos**. Os atributos A na relação r são chave estrangeira referenciando os atributos B na relação se e só se: $\Pi_A(r) \subseteq \Pi_B(s)$.

Propriedades da projeção:

- *Comutatividade* — $\Pi_{C_n}(\sigma_p(r)) = \sigma_p(\Pi_{C_n}(r))$.

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

Table 3: Relação r

A	C
α	1
β	1
β	2

Table 4: $\Pi_{A,C}(r)$

3.3 Operação de união

$$r \cup s = \{t \mid t \in r \vee t \in s\}$$

Para $r \cup s$ ser válida, r e s devem ter o mesmo número de atributos e os domínios dos atributos devem ser compatíveis (ou seja os valores da 2a coluna de r são do mesmo tipo dos valores da 2a coluna de s).
Propriedade da União:

- *Comutatividade*: $r \cup s = s \cup r$.
- *Associatividade*: $r \cup (s \cup t) = (r \cup s) \cup t$
- *Distributividade sobre seleção*: $\sigma_p(r \cup s) = \sigma_p(r) \cup \sigma_p(s)$.
- *Distributividade sobre projeção*: $\Pi_L r \cup s = \Pi_L(r) \cup \Pi_L(s)$.

A	B
α	1
α	2
β	1

Table 5: Relação r

A	B
α	2
β	3

Table 6: Relação s

A	B
α	1
α	2
β	1
β	3

Table 7: $r \cup s$

3.4 Operação de diferença de conjuntos

$$r - s = \{t \mid t \in r \wedge t \notin s\}$$

As diferenças de conjuntos só podem ser efetuadas entre relações compatíveis. Para tal r e s devem ter o mesmo número de elementos e os domínios dos atributos de r e s devem ser compatíveis.
Propriedade da diferença:

- *Distributividade sobre seleção*: $\sigma_p(r - s) = \sigma_p(r) - \sigma_p(s)$.

A	B
α	1
α	2
β	1

Table 8: Relação r

A	B
α	2
β	3

Table 9: Relação s

A	B
α	1
β	1

Table 10: $r - s$

3.5 Operação de produto cartesiano

$$r \times s = \{tq \mid t \in r \wedge q \in s\}$$

Assume que os atributos de $r(R)$ e $s(S)$ são disjuntos. (Ou seja, $R \cap S = \varnothing$). **Se os atributos de $r(R)$ e $s(S)$ não são disjuntos, então têm que se utilizar renomeações**. Se o nome das relações for diferente, distinguem-se os atributos com o mesmo nome prefixando-os com o nome da relação.
Propriedades do Produto Cartesiano:

- **Associatividade**: $r \times (s \times t) = (r \times s) \times t$.
- **Não é Comutativo**.

A	B
α	1
β	2

Table 11: Relação r

C	D	E
α	10	a
α	13	a
β	20	b
γ	10	b

Table 12: Relação s

A	B	C	D	E
α	1	α	10	a
α	1	α	13	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	α	13	a
β	2	β	20	b
β	2	γ	10	b

Table 13: $r \times s$

3.6 Operação de renomeação

$$\rho_X(E)$$

Devolve a expressão E com o nome x. Permite dar um nome aos resultados de expressões de álgebra relacional e que uma relação seja referida por mais de um nome. Se uma expressão de álgebra relacional E tem número de elementos n, então $\rho_{X(A_1,A_2,\ldots,A_n)}(E)$ devolve a expressão E com o nome X, e com os atributos renomeados para $(A_1, A_2, \ldots, A_n)$ respetivamente.

3.7 Operação de interseção de conjuntos

$$r \cap s = \{t \mid t \in r \wedge t \in s\}$$

Obriga a que r e s tenham o mesmo número de elementos (atributos) e a que os atributos de r e s sejam compatíveis (do mesmo tipo).

A	B
α	1
α	2
β	1

Table 14: Relação r

A	B
α	2
β	3

Table 15: Relação s

A	B
α	2

Table 16: $r \cap s$

3.8 Operação de junção natural

$$r \bowtie s$$

Sejam r e s relações nos esquemas R e S, respetivamente. O resultado é uma relação no esquema $R \cup S$ que é obtido considerando cada par de tuplos t_r de r e t_s de s. Se t_r e t_s têm o mesmo valor em cada um dos atributos de $R \cap S$, então um tuplo t é adicionado ao resultado, em que t tem o mesmo valor que t_r em r e t tem o mesmo valor que t_s em s.

$$R = (A, B, C, D) \quad S = (E, B, D) \quad r \bowtie s = (A, B, C, D, E)$$
$$r \bowtie s = \Pi_{r.B,r.D,r.A,r.C,s.E}(\sigma_{r.B=s.B \ \wedge \ r.D=s.D}(r \times s))$$

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

Table 17: Relação r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

Table 18: Relação s

B	D	A	C	E
1	a	α	α	α
1	a	α	α	γ
1	a	α	γ	α
1	a	α	γ	γ
2	b	δ	β	δ

Table 19: $r \bowtie s$

3.9 Operação de divisão

$$r \div s = \{t \mid t \in \Pi_{R-S}(r) \ \wedge \ \forall_u \in s : tu \in r \ \}$$

Onde tu é o tuplo resultante da concatenação dos tuplos t e u. Adequada para consultas que incluam a frase **"para todo"**. Sejam r e s relações nos esquemas R e S respetivamente, com $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$ e $S = (B_1, \ldots, B_n)$, Portanto $R - S = (A1, ..., Am)$. Seja $q = r \div s$, então q é a maior relação satisfazendo $q \times s \subseteq r$. Definição em termos de operações básicas da álgebra relacional. Sejam $r(R)$ e $s(S)$ relações, com $S \subseteq R$:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - r)$$

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

Table 20: Relação r

B
1
2

Table 21: Relação s

A
α
β

Table 22: $r \div s$

3.10 Operação de Atribuição

A operação de atribuição ($\leftarrow$) permite-nos renomear consultas complexas. Escreve-se a consulta como um programa sequencial constituído por uma sequência de atribuições terminada com uma expressão cujo valor é o resultado da consulta. **A atribuição é sempre efetuada para uma variável de relação temporária**. O resultado à direita de $\leftarrow$ é atribuído à variável que se encontra à esquerda de $\leftarrow$. Ou seja podemos escrever $r \div s$ como:

$$temp_1 \leftarrow \Pi_{R-S}(r) \quad temp_2 \leftarrow \Pi_{R-S}((temp_1 \times s) - r) \quad resultado \leftarrow temp_1 - temp_2$$

3.11 Projeção Generalizada

$$\Pi_{F_1,F_2,\ldots,F_n}(E)$$

E é um expressão arbitrária. Cada uma das expressões $(F_1, F_2, \ldots, F_n)$ é uma expressão aritmética envolvendo constantes e atributos no esquema de E. Exemplo: Dada a relação creditInfo*(customerName, limit, creditBalance)*, encontrar quanto cada cliente pode ainda gastar: $\Pi_{customerName,limit-creditBalance}$(creditInfo).

3.12 Funções de Agregação

Funções de Agregação

Aplicam-se a uma coleção de valores e devolvem um único valor como resultado.

- **avg**: média dos valores.
- **min**: minimo dos valores.
- **max**: máximo dos valores.
- **sum**: soma dos valores.
- **count**: número de valores distintos.

Operação de Agregação

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(E)$$

E é um expressão. $(G_1, G_2, \ldots, G_n)$ é uma lista de atributos de agrupamento (pode ser vazia). Cada F_i é uma função de agregação e cada A_i é um nome de um atributo. Agrupa a relação resultado de E em conjuntos que tenham valores iguais nos atributos $(G_1, G_2, \ldots, G_n)$ (se $n = 0$, faz um grupo com todo o E). **Para cada grupo, devolve um tuplo**. Esse tuplo tem os valores dos atributos $(G_1, G_2, \ldots, G_n)$ do grupo, mais o resultado de aplicar as várias $F_i(Ai)$ ao conjunto de valores do grupo. O resultado da agregação não tem um nome. Pode-se recorrer à operação de renomeação para lhe dar um nome. Exemplo: $_{branchName}\mathcal{G}_{sum(balance)}$ **as** $_{sumBalance}(account)$.

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

Table 23: Relação r

sum(C)
27

Table 24: $\mathcal{G}_{sum(C)}(r)$

3.13 Junção Externa

Uma extensão da operação de junção que evita a perda de informação. Calcula a junção e depois adiciona ao resultado os tuplos de uma relação que não estão relacionados com a outra relação na junção. Utiliza valores nulos. Todas as comparações com null são falsas por definição.

Falta a remoção inserção e atualização.

4 Structured Query Language - SQL

Um comando SQL genérico

```
select F_agr1, ... F_agrn
from R_1, ..., R_m
where Cond_W
group by A_1, ..., A_g
having Cond_H
```

4.1 Select

A cláusula select corresponde à operação de projeção da álgebra relacional. É utilizada para listar os atributos pretendidos no resultado da consulta. Na sintaxe de álgebra relacional a consulta seria: $\Pi_{atributo}(tabela)$. Um asterisco na cláusula select denota "todos os atributos" (elimina a projeção).

```
select atributo
from tabela
```

O SQL permite duplicados nas relações e nos resultados de consultas. Para forçar a eliminação de duplicados no resultado, inserir a palavra-chave distinct após select.

```
select distinct atributo
from tabela
```

A palavra-chave all indica que os duplicados não devem ser removidos (por omissão, o all é assumido pelo sistema, mesmo que não o coloque lá).

A cláusula select pode conter expressões aritméticas envolvendo as operações, $+, -, *, e/$, com argumentos constantes ou atributos de tuplos (como na projeção generalizada).

```
select atributo_1, atributo_2 * 100
from tabela
```

Devolve um relação anterior, exceto que o $atributo_2$ é multiplicado por 100.

4.2 Where

A cláusula where corresponde ao predicado de seleção da álgebra relacional. É formada por um predicado envolvendo atributos de relações que aparecem na cláusula from.

```
select atributo
from tabela
where atributo > ...
```

Os resultados de comparações podem ser combinados por intermédio dos conectivos lógicos *and*, *or*, e *not*. Podem-se aplicar comparações ao resultado de expressões aritméticas. Operador de comparação between para especificar condições em que um valor deve estar contido num intervalo de valores (incluindo os seus extremos). Para negar a condição pode-se colocar o conetivo not antes de between.

```
select atributo
from tabela
where val1 between val2 and val3
```

Também existe o operador in para testar se um valor pertence a uma lista.

```
select atributo
from tabela
where val1 in ('val2', 'val3', 'val4')
```

4.3 Cadeias de Caracteres

O SQL inclui um mecanismo de concordância de padrões para comparações envolvendo cadeias de caracteres. Os padrões são descritos recorrendo a dois caracteres especiais:

- percentagem(%) - O carácter % concorda com qualquer subcadeia.
- sublinhado (_) - O carácter _ concorda com qualquer carácter.

```
select atributo
from tabela
where val1 like '%...%'
```

A SQL suporta uma variedade de operações com cadeias de caracteres, tais como:

- concatenação (utilizando ||)
- conversão para maiúsculas (upper) e para minúsculas (lower)
- calcular o comprimento, extração de subcadeias, etc.

4.4 From

A cláusula from corresponde à operação de produto cartesiano da álgebra relacional. Indica as relações a consultar na avaliação da expressão.

```
select *
from tabela1 , tabela2 --> produto cartesiano entre tabela1 e tabela2.
```

4.5 Renomeação

A linguagem SQL permite a renomeação de relações e atributos recorrendo à cláusula **as**:

```
old_name as new_name
```

Caso se pretenda utilizar um nome com espaços, esse nome deverá ser colocado entre aspas.

4.6 Variáveis de tuplo

As variáveis de tuplo são definidas na cláusula from por intermédio da cláusula opcional **as**:

```
select atributo
from tabela1 as T, tabela2 as S
```

As variáveis de tuplos podem ser vistas como criando várias cópias de uma mesma relação.

```
--> voos(numVoo,Matr,Data,Hora,De,Para) Quais os pares de voos que usaram o mesmo aviao no
    mesmo dia?
select T.numVoo, S.numVoo
from voos as T, voos as S
where T.Matr = S.Matr
  and T.Data = S.Data
  and T.numVoo < S.numVoo
```

4.7 Select Case

O **case** pode ocorrer em qualquer lugar onde se pode ter uma expressão por exemplo na cláusula select, where ou mesmo dentro de funções de agregação.

```
select numero,
case when sexo='M' then 'Senhor' || nome
     when sexo='F' then 'Senhora' || nome
  end as titNome
from alunos
```

Pode-se ter um ramo else final:

```
select alu_numero as aluno, cad_codigo as cadeira,
case when nota='X' then 'Excluido'
     when nota='A' then 'Ausente '
     when nota='F' then 'Faltou'
     when nota is null then 'Nota por lancar'
     else nota
end
from inscricoes
```

4.8 Ordenação de tuplos

```
select numero, name
from alunos
order by name
```

Pode-se especificar desc para ordenação descente ou asc para ordenação ascendente, para cada atributo; por omissão, assume-se ordem ascendente.

```
select numero, name
from alunos
order by name desc
```

Pode-se especificar *nulls first* ou *nulls last*, após o desc/asc, para indicar como ordenar os valores nulos. **Pode-se ter mais do que uma chave de ordenação, separando-as com vírgulas**.

4.9 Operações com Conjuntos

As operações com conjuntos union, intersect, e except (minus no Oracle) operam sobre relações e correspondem aos operadores de álgebra relacional $\cup, \cap$ e $-$ respetivamente. **Ao contrário das outras operações em SQL, as operações com conjuntos eliminam os duplicados automaticamente**. Para reter duplicados deve-se utilizar as respectivas versões multiconjunto union all, intersect all e except all. Supondo que um tuplo ocorre m vezes em r e n vezes em s, então ele ocorre:

- $m + n$ vezes em r union all s.
- $min(m, n)$ vezes em r intersect all s.
- $max(0, m–n)$ vezes em r except all s.

```
--> Listar todos os clientes que tem um emprestimo ou uma conta:
(select customer_name from depositor)
union
(select customer_name from borrower)
--> Listar todos os clientes que tem um emprestimo e uma conta:
(select customer_name from depositor)
intersect
(select customer_name from borrower)
--> Listar todos os clientes que tem uma conta mas nao tem emprestimos:
(select customer_name from depositor)
except
(select customer_name from borrower)
```

4.10 Funções de Agregação

Estas funções aplicam-se a multi-conjuntos de valores de uma coluna de uma relação, devolvendo um valor:

- **avg**: valor médio
- **min**: valor mínimo
- **max**: valor máximo
- **sum**: soma dos valores
- **count**: número de valores

```
--> Determinar o saldo medio das contas
select avg (balance)
from account
where holder_name = bpi

--> Calcular o numero de clientes.
select count (*)
from customer

--> Encontrar o numero de depositantes do banco.
select count (distinct customer_name)
from depositor
```

4.11 Group By

Atributos na cláusula **select** fora de funções de agregação têm de aparecer na lista **group by**. Se aparecer mais do que um atributo em group by, então cada grupo é formado pelos tuplo com a mesma combinação de valores em todos esses os atributos

```
--> Listar o numero de depositantes por agencia.
select branch_name, count (distinct customer_name)
from depositor, account
where depositor.account_number = account.account_number
group by branch_name
```

4.12 Having

Predicados na cláusula **having** são aplicados depois da formação dos grupos, enquanto que os predicados na cláusula **where** são aplicados antes da formação dos grupos.

```
--> Listar os nomes de todas as agencias cujo valor medio dos saldos das contas superior a
    1,200.

select branch_name, avg (balance)
from account
group by branch_name
having avg (balance) > 1200
```

4.13 Operações de Junção

Estas operações adicionais são utilizadas habitualmente em sub-consultas na cláusula **from**.

- **Condição de junção**: define quais os tuplos que são combinados nas duas relações, assim como quais os atributos que aparecem no resultado da junção.
- **Tipo de junção**: define como tratar os tuplos que não estão relacionados entre si (baseados na condição de junção).
  - **inner join**.
  - **left outer join**.
  - **right outer join**.
  - **full outer join**.

```
--> Condicoes de juncao
natural
on <predicate>
using (A_1, A_2, ..., A_n)
```

- É obrigatória a utilização de uma condição de junção nas junções de tipo **outer**. É opcional no **inner join** (na ausência, comporta-se como o produto cartesiano).
- A condição natural aparece antes do tipo de junção (por exemplo **natural inner join**). As restantes condições aparecem após o tipo de junção.
- Nas junções com condição **natural**, primeiro aparecem os atributos comuns a ambas as relações, na ordem pela qual aparecem na relação do lado esquerdo. Depois, aparecem os restantes atributos da relação do lado esquerdo, seguidos dos restantes atributos da relação do lado direito.
- Nas junções com condição **using** $(A_1, A_2, \ldots, A_n)$, primeiro aparecem os atributos $A_1, A_2, \ldots, A_n$. Depois, aparecem os restantes atributos da relação do lado esquerdo, seguidos dos restantes atributos da relação do lado direito.

4.14 Using

```
--> Listar todos os clientes que tem uma conta e nenhum emprestimo
select customer_name
from depositor left outer join borrower using(customer_name)
where loan_number is null

--> Listar todos os clientes que tem uma conta ou um emprestimo no banco (mas nao ambos)
select customer_name
from depositor natural full outer join borrower
where account_number is null or loan_number is null
```